Collective Communication in Recursive Dual-Net: An Interconnection Network for High-Performance Parallel Computer Systems of the Next Generation

Yamin Li and Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{*yamin, speng*}*@k.hosei.ac.jp*

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

***Abstract*—In this paper, we propose efficient routing algorithms for collective communication in a newly proposed, versatile network, called a recursive dual-net (RDN). The RDN can be used as a candidate for the interconnection of supercomputers of the next generation. The RDN is generated by recursively applying dual-construction on a base network. Given a regular and symmetric graph of size n and node-degree d, the dual-construction generates a regular and symmetric graph of size $2n^2$ and node-degree $d+1$. The RDN has many interesting properties including small node-degree and short diameter. Our results show that collective communication can be done efficiently in RDN.**

Key words: Interconnection networks, collective communication, routing algorithms

I. Introduction

In massively parallel processor (MPP), the interconnection network plays a crucial role in the issues such as communication performance, hardware cost, computational complexity, fault-tolerance. Much research has been reported in the literature for interconnection networks that can be used to connect parallel computers of large scale (see [1], [2], [3] for the review of the early work). The following two categories have attracted a great research attention. One is the networks of hypercube-like family that has the advantage of short diameters for high-performance computing and efficient communication [4], [5], [6], [7], [8]. The other is 2D/3D meshes or tori that has the advantage of small and fixed node-degrees and easy implementations. Traditionally, the MPP in the history including those built by NASA, CRAY, FGPS, IBM, use a 2D/3D mesh or a torus or their variations with extra diagonal links. Recursive networks also have been proposed as effective interconnection networks for parallel computers of large scale. For example, the WK-recursive network [9], [10] is a class of recursive scalable networks. It offers a high-degree of regularity, scalability, and symmetry and has a compact VLSI implementation.

Recently, due to advances in computer technology, and competition among computer makers, computers containing hundreds of thousands of nodes have been built [11]. It was predicted that the MPPs of the next decade will contain 10 to 100 millions of nodes [12]. For such a parallel computer of very-large scale, the traditional interconnection networks may no longer satisfy the requirements for the high-performance computing or efficient communication. For the future generation of MPPs with millions of nodes, the node-degree and the diameter will be the critical measures for the effectiveness of the interconnection networks. The node-degree is limited by the hardware technologies and the diameter affects all kind of communication schemes directly. Other important measures include bisection bandwidth, scalability, and efficient routing algorithms.

In this paper, we first describe a set of networks, called *Recursive Dual-Net* (RDN). The RDN is based on recursive dual-construction of a base network. The dual-construction extends a regular network with n nodes and node-degree d to a regular network with $2n^2$ nodes and node-degree $d+1$. The RDN is especially suitable for the interconnection of parallel computers with millions of nodes. It can connect a huge number of nodes with just a small number of links per node and very short diameters. For example, a 2-level RDN with 5-ary, 2-cube as the base network can connect more than 3-million nodes with only 6 links per node and its diameter equals to 22. The RDN extends the idea of the second level interconnection network of IBM Roadrunner in which every cluster is connected to every switch [13].

The major contributions of this paper are the efficient routing algorithms for collective communication including one-to-all broadcast, all-to-all broadcast, one-to-all personalized communication, and all-to-all personalized communication. The proposed algorithms are analyzed carefully under 1-port, package switching communication model. Then, the results are compared with hypercube networks.

The rest of this paper is organized as follows. Section 2 describes the recursive dual-net in details. Section 3 describes the communication model used in this paper. Section 4 describes the routing algorithms for one-to-all broadcast and personalized communication. Sections 5 describes the routing algorithms for all-to-all broadcast and personalized communication. Section 6 compares the time complexities of

the proposed algorithms and those of hypercube. 7 concludes the paper and presents some future research directions.

II. Recursive Dual-Net

Let G be an undirected graph. The size of G, denoted as $|G|$, is the number of vertices. A path from node s to node t in G is denoted by $s \rightarrow t$. The length of the path is the number of edges in the path. For any two nodes s and t in G, we denote $L(s,t)$ as the length of a shortest path connecting s and t. The diameter of G is defined as $D(G) = \max\{L(s,t)|s,t \in G\}$. For any two nodes s and t in G, if there is a path connecting s and t, we say G is a connected graph.

Suppose we have a regular graph B and there are n_0 nodes in B and the node-degree is d_0. A k-level Recursive Dual-Net $RDN^k(B)$, also denoted as $RDN^k(B(n_0))$, can be recursively defined as follows:

1) $RDN^0(B) = B$ is a regular graph with n_0 nodes, called *base network*;
2) For $k > 0$, $RDN^k(B)$ is constructed from $RDN^{k-1}(B)$ by a dual-construction as explained below (also see Figure 1).

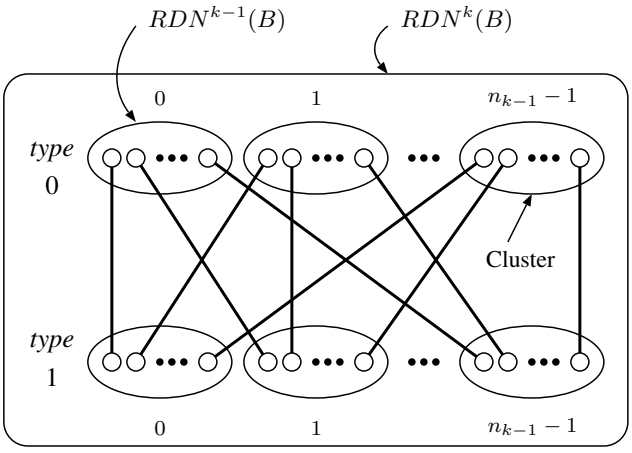


Figure 1. Build an $RDN^k(B)$ from $RDN^{k-1}(B)$

Dual-construction: Let $RDN^{k-1}(B)$ be referred to as a *cluster* of level k and $n_{k-1} = |RDN^{k-1}(B)|$ for $k > 0$. An $RDN^k(B)$ is a graph that contains $2n_{k-1}$ clusters of level k as subgraphs. These clusters are divided into two sets with each set containing n_{k-1} clusters. Each cluster in one set is said to be of *type* 0, denoted as C_i^0, where $0 \le i \le n_{k-1}-1$ is the cluster ID. Each cluster in the other set is of *type* 1, denoted as C_j^1, where $0 \le j \le n_{k-1}-1$ is the cluster ID. At level k, each node in a cluster has a new link to a node in a distinct cluster of the other type. We call this link *cross-edge* of level k. By following this rule, for each pair of clusters C_i^0 and C_j^1, there is a unique edge connecting a node in C_i^0 and a node in C_j^1, $0 \le i,j \le n_{k-1}-1$. In Figure 1, there are n_{k-1} nodes within each cluster $RDN^{k-1}(B)$.

We can see from the recursive dual-construction described above that an $RDN^k(B)$ is a symmetric regular network with node-degree $d_0 + k$ if the base network is a symmetric regular network with node-degree d_0. The following theorem is from [14].

Theorem 1: Assume that the base network B is a symmetric graph with size n_0, node-degree d_0, and the diameter D_0. Then, the size, the node-degree, the diameter and the bisection bandwidth of $RDN^k(B)$ are $(2n_0)^{2^k}/2$, $d_0 + k$, $2^k D_0 + 2^{k+1} - 2$, and $\lceil (2n_0)^{2^k}/8 \rceil$, respectively.

The *cost ratio* $CR(G)$ for measuring the combined effects of the hardware cost and the software efficiency of an interconnection network was also proposed in [14]. Let $|(G)|$, $d(G)$, and $D(G)$ be the number of nodes, the node-degree, and the diameter of G, respectively. We define $CR(G)$ as

$$CR(G) = (d(G) + D(G))/\lg|(G)|$$

The cost ratio of an n-cube is 2 regardless of its size. The CR for some $RDN^k(B)$ is shown in Table I. Two small networks including 3-ary 3-cube and 5-ary 2-cube are selected as practical base networks. For INs of size around 1K, we set $k = 1$, while for INs of size larger 1M, we set $k = 2$. The results show that the cost ratios of $RDN^k(B)$ are better than hypercube and 3D-tori in all cases.

Table I
CR for some $RDN^k(B)$

Network	n	d	D	CR
10-cube	1,024	10	10	2.00
$RDN^1(B(25))$	1,250	5	10	1.46
$RDN^1(B(27))$	1,458	7	8	1.43
3D-Tori(10)	1,000	6	15	2.11
22-cube	4,194,304	22	22	2.00
$RDN^2(B(25))$	3,125,000	6	22	1.30
$RDN^2(B(27))$	4,251,528	8	18	1.18
3D-Tori(160)	4,096,000	6	240	11.20

A presentation for $RDN^k(B)$ that provides an unique ID to each node in $RDN^k(B)$ is described as follows. Let the IDs of nodes in B, denoted as ID_0, be i, $0 \le i \le n_0 - 1$. The ID_k of node u in $RDN^k(B)$ for $k > 0$ is a triple (u_0, u_1, u_2), where u_0 is a 0 or 1, u_1 and u_2 belong to ID_{k-1}. We call u_0, u_1, and u_2 typeID, clusterID, and nodeID of u, respectively.

More specifically, ID_i, $1 \le i \le k$, can be defined recursively as follows: $ID_i = (b, ID_{i-1}, ID_{i-1})$, where $b = 0$ or 1. The ID of a node u in $RDN^k(B)$ can also be presented by an unique integer i, $0 \le i \le (2n_0)^{2^k}/2 - 1$, where i is the lexicographical order of the triple (u_0, u_1, u_2). For example, the ID of node $(1,1,2)$ in $RDN^1(B)$ is $1 * 3^2 + 1 * 3 + 2 = 14$. With this ID presentation, (u, v) is

a cross-edge of level k in $RDN^k(B)$ iff $u_0 \neq v_0$, $u_1 = v_2$, and $u_2 = v_1$. The following theorem is from [14].

Theorem 2: In $RDN^k(B)$, routing from source s to destination t can be done in at most $2^k * D_0 + 2^{k+1} - 2$ steps, where D_0 is the diameter of the base network.

III. Model of Communications

Design of efficient routing algorithms for collective communications is the key issue in message-passing parallel computers or networks [15] [16] [17] [18] [19]. Collective communications are required in load balancing, event synchronization, and data exchange. Based on the number of sending and receiving processors, these communications can be classified into one-to-many, one-to-all, many-to-many and all-to-all. The nature of the messages to be sent can be classified as personalized or non-personalized (multicast or broadcast). The all-to-all personalized communication (total exchange) is at the heart of numerical applications. In this paper, we will present efficient algorithms for collective communications in recursive dual-net.

An important metric used to evaluate efficiency of communication is *transmission latency*, or *communication time*. The communication time depends on many factors such as contentions, switching techniques, network topologies etc. Therefore, we first define the communication model used in this paper.

We assume that the communication links are bidirectional, that is, two directly-connected processors can send messages to each other simultaneously. We also assume the processor-bounded model (one-port model) in which each node can access the network through a single input port and a single output port at a time. The port model of a network system refers to the number of internal channels at each node. In order to reduce the complexity of communication hardware, many systems support one-port communication architecture. We also assume the linear cost model [20] in which the transfer time for a message is linearly proportional to the length of the message.

There are many switching methods. In this paper, we assume the packet switching model [21], [22]. In this model, each packet is maintained as an entity that is passed from node to node as it moves through the network. The long message can be partitioned and transmitted as fixed-length word w. The first few bytes of a packet contains routing and control information and are referred as packet header. A packet is completely buffered at each intermediate node before it is forwarded to the next node (for this reason, the model is also called store-and-forward switching). In this paper, we allow packages that are headed for the same destination to be combined into a single message. The time to pack and unpack messages is included in the startup latency. The packet switching model is suitable for collective communication in MPP since it is safer than other switching models such as virtual cut-through switching. With packet switching model, the communication time for a message of length m (number of fixed-length words) to be sent to a node of distance d is $d(t_s + mt_w)$, where t_s is startup latency, the time required for the system to handle the message at the sending node, t_w is the per-word transfer time ($1/t_w$ is the bandwidth of the communication links). Through this paper, we will use the formula above for estimating the communication times of the proposed algorithms.

IV. One-to-All Broadcast and Personalized Communication

In this section, we discuss one-to-all broadcast and one-to-all personalized communication.

Parallel algorithms often require a single processor to send identical data to all other processors or to a subset of them. This operation is known as one-to-all broadcast or one-to-many multicast. In our communication model, a message is not routed in parts along separate paths and communication is allowed on only one link of each processor at a time. It can be shown that one-to-all broadcast cannot be performed in less than $(\log p)(t_s + mt_w)$ time on any architecture, where p is the number of processors [22].

We show an algorithm which performs one-to-all broadcast in RDN efficiently. The algorithm for broadcast from source node s works as follows. The source s first sends the message to it neighbor s' along the cross-edge of level k. Then, s and s' broadcast simultaneously the message to all other nodes in C_s and $C_{s'}$ recursively. Next, every node $u \in C_s \setminus \{s\}$ and every node $u' \in C_{s'} \setminus \{s'\}$ send the message to neighbors v and v' along the cross-edge of level k, respectively. Finally, every v and v' broadcast the message to all other nodes in C_v and $C_{v'}$, respectively. The algorithm is formally shown in Algorithm 1. All nodes execute it concurrently.

Algorithm 1: One_To_All_Bcast($RDN^k(B), my\_id, s, msg$)
begin
  if $k = 0$ One_To_All_Bcast(B, my_id, s, msg)
  else
    $partner \leftarrow$ the neighbor of my_id via the cross-edge of level k;
    if $my\_id = s$ **send** msg to $partner$;
    if ($my\_id = s$) OR ($partner = s$)
      $source \leftarrow my\_node\_id$;
      One_To_All_Bcast($RDN^{k-1}_{my\_id}(B), *, source, msg$);
    if ($my\_cluster\_id = s_1$) AND ($my\_node\_id \neq s_2$)
    /* $s = (s_0, s_1, s_2)$ */
      send msg to $partner$;
    if $getmessage = true$
      $source \leftarrow my\_node\_id$;
      One_To_All_Bcast($RDN^{k-1}_{my\_id}(B), *, source, msg$);
end

The time complexity for the one-to-all broadcasting $T_{oab}(k,m), k > 0$, can be calculated from the following recurrence:

$$T_{oab}(k,m) = 2(t_s + mt_w) + 2T_{oab}(k-1,m).$$

Assume that $T_{oab}(0,m)$ is given. Then, we can obtain the solution of the recurrence as follows.

$T_{oab}(k,m) = (2 + 2^2 + \ldots + 2^k)(t_s + mt_w) + 2^k T_{oab}(0,m) = (2^{k+1} - 2)(t_s + mt_w) + 2^k T_{oab}(0,m)$.

Theorem 3: Assume that the time complexity $T_{oab}(0,m)$ for one-to-all broadcasting in the base network B is known, where m is the length of the message. The time complexity $T_{oab}(k,m)$ for one-to-all broadcasting in $RDN^k(B), k > 0$, is $(2^{k+1} - 2)(t_s + mt_w) + 2^k T_{oab}(0,m)$.

In one-to-all personalized communication, a single node sends a unique message of size m to every other node. Since the source node transmits m words for each of the other $p-1$ nodes, the lower bound to the communication time of one-to-all personalized communication is $(p-1)mt_w$. This lower bound is independent of the architecture or routing scheme. The routing algorithm for the one-to-all personalized communication in RDN is similar to the one-to-all broadcast algorithm described above. However, in order to achieve a better performance on the communication time, we pack and unpack the messages in a proper way.

Initially, the source node s contains all the messages. First, s packs the messages for the nodes in the other level-k clusters of the same type as s into a single message $msg1$, packs the messages for the level-k clusters of the other type except $C(s')$ ($s' = s^{(k+m)}$) into a single message $msg2$ and sends $msg1$ to s' along the cross-edge of s. Second, s and s' simultaneously unpack properly messages $msg2$ and $msg1$ and send each corresponding parts to all other nodes in $C(s)$ and $C(s')$, respectively. This stage is done recursively. At the end of this stage, each $u \in C(s)$ and each $u' \in C(s') \setminus \{s'\}$ hold the packed messages of size $|C_k|m$ (say, $msgC_v$ and $msgC_{v'}$) for the nodes in level-k clusters $C(v)$ and $C(v')$, respectively, where $v = u^{(k+m)}$ and $v' = (u')^{(k+m)}$. Node s holds packed messages $msgC_s$ and $msgC_{s'}$. Third, u and u' send $msgC_v$ and $msgC_{v'}$ to v and v' (node s sends $msgC_{s'}$ to s') along cross-edges of u and u', respectively. At the end of this stage, every cluster has one node, say src for simplicity, which holds message $msgC_{src}$. Finally, node src sends the message $msgC_{src}$ to all other nodes in $C(src)$ recursively. The algorithm is formally described in Algorithm 2.

The time complexity for the one-to-all personalized communication T_{oap} is calculated as follows. Assume that the time for pack/unpack process is included in t_s. Then we have

Algorithm 2: One_To_All_Pers($RDN^k(B), my\_id, s, msg$)
begin
 if $k = 0$ One_To_All_Pers($B, , my\_id, s, M_{my\_id}$)
 else
  divide msg into two parts, $msg1$ and $msg2$, where $msg1$ contains all messages to be sent to the nodes in the clusters of type s_0, and $msg2$ contains the rest messages;
  pack all messages in $msg1$ and $msg2$ that are to be sent to the nodes in the cluster C_q with cluster_ID = q into a single message msg_q to be sent to the node with node_ID = q in the clusters with cluster_ID = s_1;
  $partner \leftarrow$ the neighbor of my_id via the cross-edge of level k;
  if $my\_id = s$
   send $msg1 - \{msgC_s\}$ to $partner$;
   $temp \leftarrow msg2 \setminus \{msgC_{s'}\}$
   One_To_All_Pers($RDN_s^{k-1}(B), *, s_2, temp$);
  if $partner = s$
   get $msg1 - \{msgC_s\}$ from $partner$;
   $temp \leftarrow msg1 \setminus \{msgC_s\}$
   One_To_All_Pers($RDN_{my\_id}^{k-1}(B), *, s_2, temp$);
  if ($my\_cluster\_id = s_1$) AND ($my\_id \neq s'$)
   send $msgC_{my\_node\_id}$ to $partner$;
  if $getmessage = true$
   get $temp$ from $partner$;
   $source \leftarrow my\_node\_id$;
  if $my\_id = s$
   $temp \leftarrow msgC_s$;
   $source \leftarrow my\_node\_id$;
   One_To_All_Pers($RDN_{my\_id}^{k-1}(B), *, source, temp$);
 endif
end

$T_{oap}(k,m) = (t_s + (n_k/2 - n_{k-1})mt_w) + T_{oap}(k-1, n_{k-1}m) + (t_s + n_{k-1}mt_w) + T_{oap}(k-1,m)$.

Assume that $T_{oap}(0,m)$ is given and $k > 0$. Then, we can obtain an approximate solution of the recurrence as follows.

$T_{oap}(k,m) \approx (2 + 2^2 + \ldots + 2^k)t_s + (n_k/2 + n_k/2^2 + \ldots)mt_w + T_{oap}(0, n_{k-1}n_{k-2}\ldots n_0 m) + T_{oap}(0, n_{k-1}n_{k-2}\ldots n_1 m) + (2^k - 2)T_{oap}(0,m) \approx (2^{k+1} - 2)t_s + n_k mt_w + T_{oap}(0, rm) + T_{oap}(0, (r/n_0)m) + (2^k - 2)T_{oap}(0,m)$,
where $r = n_k/(2^k n_0) = (2n_0)^{2^k - 1}/2^k$.

Theorem 4: Assume that the time complexity $T_{oap}(0,m)$ for one-to-all personalized communication in the base network B is known, where m is the length of each message. The time complexity $T_{oap}(k,m)$ for one-to-all personalized communication in $RDN^k(B), k > 0$, is approximately

$(2^{k+1}-2)t_s + n_k m t_w + T_{oap}(0, rm) + T_{oap}(0, (r/n_0)m) + (2^k - 2)T_{oap}(0, m)$, where $r = n_k/(2^k n_0) = (2n_0)^{2^k-1}/2^k$.

V. All-to-all Broadcast and personalized communication

All-to-all broadcast is a generalization of one-to-all broadcast in which all nodes simultaneously initiate a broadcast. A node broadcasts the same m-word message to every other node, but different nodes may broadcast different messages. The communication pattern of all-to-all broadcast can be used to perform some other operations, such as reduction and prefix sums.

The lower bound to the communication time of all-to-all broadcast for parallel computers on which a node can communicate on only one of its ports at a time is $(p-1)mt_w$, where p is the number of nodes. This is because each node receives at least $(p-1)m$ words of data, regardless of the architecture or routing scheme.

An efficient way to perform all-to-all broadcast is to perform all p one-to-all broadcasts simultaneously so that all messages traversing the same path at the same time are concatenated into a single message whose size is the sum of the sizes of individual messages.

The algorithm for all-to-all broadcast in RDN can be described in three stages. In the first stage, the broadcast is done within each level-k cluster. This procedure is done recursively. In the second stage, each node of a level-k cluster sends the identical message to a node in a level-k cluster of the other type along the level-k cross-edge, and then, the received message is broadcasted within the cluster. After this stage, every node receives messages from all other nodes in the same clusters and in the clusters of the other type. In the last stage, every node gets a single concatenated message, including the messages from the nodes in other level-k clusters of the same type, along the level-k cross-edge.

The algorithm is showed in Algorithm 3. All nodes execute the algorithm concurrently. In Algorithm 3, my_id is the id of the node. The initial message to be broadcasted is M_{my_id} at each node. At the end of the procedure, each node stores the collection of all p messages in $result$. At each stage, the algorithm first assigns $partner$ and then, performs data exchange by sending and receiving message to and from the partner. The partner for data exchange is one of its neighbors through the cross-edge of level k.

The time for the all-to-all broadcast T_{aab} is calculated as follows.

$$T_{aab}(k, m) = T_{aab}(k-1, m) + (t_s + n_{k-1} m t_w) + T_{aab}(k-1, n_{k-1}m) + (t_s + (n_{k-1}^2) m t_w)$$

We can see from the above formula that the time complexities for the all-to-all broadcasting and the one-to-all personalized communication are approximately the same.

Algorithm 3: All_To_All_Bcast($RDN^k(B), M_{my\_id}$)
begin
  if $k = 0$ $result \leftarrow$ All_To_All_Bcast(B, M_{my_id})
  else
    $M'_{my\_id} \leftarrow$ All_To_All_Bcast($RDN^{k-1}(B), M_{my\_id}$);
    $partner \leftarrow$ the neighbor via cross-edge of level k;
    send message M'_{my_id} to $partner$;
    $T_{my\_id} \leftarrow M'_{partner}$;
    $T'_{my\_id} \leftarrow$ All_To_All_Bcast($RDN^{k-1}(B), T_{my\_id}$);
    send message T'_{my_id} to $partner$;
    $result \leftarrow T'_{my\_id} \cup T'_{partner}$;
  endif
end

Theorem 5: Assume that the time complexity $T_{aab}(0, m)$ for all-to-all broadcasting in the base network B is known, where m is the length of each message. The time complexity $T_{aab}(k, m)$ for all-to-all broadcasting in $RDN^k(B), k > 0$, is approximately $(2^{k+1}-2)t_s + n_k m t_w + T_{aab}(0, rm) + T_{aab}(0, (r/n_0)m) + (2^k - 2)T_{aab}(0, m)$, where $r = n_k/(2^k n_0) = (2n_0)^{2^k-1}/2^k$..

In all-to-all personalized communication, each node sends a distinct message of size m to every other node. The total number of messages is $p(p-1)$.

The algorithm for all-to-all personalized communication in RDN can be described in three stages. In the first stage, We first divide M_{my_id} into two parts, $M1_{my\_id}$ and $M2_{my\_id}$, where $M1_{my\_id}$ contains all messages to be sent to the nodes in the clusters of my_type, and $M2_{my\_id}$ contains the rest message. Then, the first part of personalized messages $M1$ is exchanged between my_id and $partner$, the neighbor via cross-edge of level k. In the second stage, we first pack all messages that are to be sent to the nodes in the cluster of level k with clusterID = q into a single message msg_q. Then, we perform all-to-all personalized communication inside each cluster, where msg_q is to be sent to node with nodeID = q. Notice that $|msg_q| = 2n_{k-1}m$. In the last stage, each node packs the received messages into a single message of length $n_k m$ and sends the packed message to its neighbor along the cross-edge of level k. After receive the message, each node unpacks the message received from its neighbor into n_{k-1} messages, $msg_{q'}$, where $msg_{q'}$ is the collection of all messages destinated to node with nodeID = q' in the cluster. Finally, we perform all-to-all personalized communication again within each cluster of level k. This can be done since the packed messages sent through the level-k cross-edge are all destinated to the nodes inside the cluster.

The algorithm is showed in Algorithm 4. All nodes execute the algorithm concurrently. In Algorithm 4, my_id is the id of the node. The initial message to be sent is

M_{my_id} which contains $p-1$ messages of length m. At the end of the algorithm, each node stores the collection of all p messages in $result$.

Algorithm 4: All_To_All_Pers($RDN^k(B), M_{my\_id}$)
begin
if $k = 0$ $result \leftarrow$ All_To_All_Pers(B, M_{my_id})
else
divide M_{my_id} into two parts, $M1_{my\_id}$ and $M2_{my\_id}$, where $M1_{my\_id}$ contains all messages to be sent to the nodes in the clusters of type my_type, and $M2_{my\_id}$ contains the rest messages;
$partner \leftarrow$ the neighbor via cross-edge of level k;
send message $M1_{my\_id}$ to $partner$;
$M'_{my\_id} \leftarrow M2_{my\_id} \cup M1_{partner}$;
pack all messages in M'_{my_id} that are to be sent to the nodes in the cluster of level k with cluster_ID = q into a single message msg_q to be sent to the node with node_ID = q;
$T_{my\_id} \leftarrow$ All_To_All_Pers($RDN^{k-1}(B), M'_{my\_id}$);
send message T_{my_id} to $partner$;
$T'_{my\_id} \leftarrow T_{partner}$;
unpack T'_{my_id} into n_{k-1} messages, $msg_{q'}$, such that $msg_{q'}$ is the collection of messages destinated to the node with node_ID = q';
$result \leftarrow$ All_To_All_Pers($RDN^{k-1}(B), T'_{my\_id}$);
endif
end

The time to complete the all-to-all personalized communication T_{aap} is as follows.

$T_{aap}(k,m) = (t_s + (n_k/2 - 1)mt_w) + T_{aap}(k - 1, 2n_{k-1}m) + (t_s + (n_k - 1)mt_w) + T_{aap}(k-1, 2n_{k-1}m) \approx 2t_s + 3n_k/2mt_w + 2T_{aap}(k-1, 2n_{k-1}m)$.

Assume that $T_{aap}(0,m)$ is given and $k > 0$. Then, we can obtain an approximate solution of the recurrence as follows.

$T_{aap}(k,m) \approx (2t_s + (3/2)n_k mt_w) + 2T_{aap}(k - 1, 2n_{k-1}m) \approx (2^{k+1} - 2)t_s + (3/2)(n_k + 2n_k + 2^2 n_k + \ldots)mt_w + 2^k T_{aap}(0, 2^k n_{k-1}n_{k-2}\ldots n_0 m) \approx (2^{k+1} - 2)t_s + (3 * 2^k * n_k)mt_w + 2^k T_{aap}(0, (n_k/n_0)m)$, where $n_k = (2n_0)^{2^k}/2$.

Theorem 6: Assume that the time complexity $T_{aap}(0,m)$ for all-to-all personalized communication in the base network B is known, where m is the length of each message. The time complexity $T_{aap}(k,m)$ for all-to-all personalized communication in $RDN^k(B), k > 0$, is approximately $(2^{k+1} - 2)t_s + (3 * 2^k * n_k)mt_w + 2^k T_{aap}(0, (n_k/n_0)m)$.

VI. Recursive Dual-net v.s. Hypercube

In this section, will compare the performance of the proposed collective communication algorithms with that of hypercube. For simplicity, we assume that the base network is a 3-cube. Then, the sizes of $RDN^1(B)$ and $RDN^2(B)$ are $2^7 = 128$ and $2^{15} = 32,768$, respectively. The time complexities of the algorithms for collective communications on hypercubes are listed below [22].

$Tcube_{oab} = (\lg p)(t_s + mt_w)$
$Tcube_{oap} = (\lg p)t_s + (p-1)mt_w$
$Tcube_{aab} = (\lg p)t_s + (p-1)mt_w$
$Tcube_{aap} = (\lg p)(t_s + (p/2)mt_w)$

For the base network, a 3-cube,

$T3cube_{oab} = 3t_s + 3mt_w$
$T3cube_{oap} = 3t_s + 7mt_w$
$T3cube_{aab} = 3t_s + 7mt_w$
$T3cube_{aap} = 3t_s + 12mt_w$

If $p = 128$, for 7-cube,

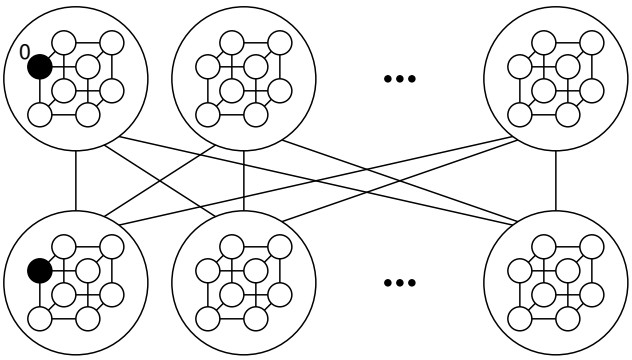
$T7cube_{oab} = 7t_s + 7mt_w$
$T7cube_{oap} = 7t_s + 127mt_w$
$T7cube_{aab} = 7t_s + 127mt_w$
$T7cube_{aap} = 7t_s + 448mt_w$

Now consider $RDN^1(B)$ where B is a 3-cube. The total number of nodes is 128.

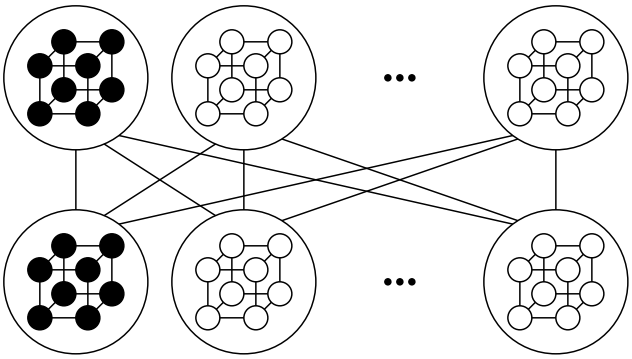
For the one-to-all broadcast, there are 4 steps as shown as in Figure 2. First, the source node s sends the message via cross-edge to the nodes s' of the other type. This takes $t_s + mt_w$. Then s and s' broadcast the message inside their own clusters (3-cubes). This takes $3t_s + 3mt_w$. Next, the nodes in the clusters of s and s' send the message via cross-edge. This takes $t_s + mt_w$. Finally, the nodes broadcast the message inside their own clusters. This takes $3t_s + 3mt_w$. The total time the one-to-all broadcast takes is $8t_s + 8mt_w$.

For the one-to-all personalized communication, there are also 4 steps. Suppose node 0 contains 128 messages, each for node i, $0 \le i \le 127$. First, the source node s packs messages for the nodes in the other clusters of the same type with s into a single message of length $(64-8)m$ in words and sends it to s' via cross-edge. It takes $t_s + (64-8)mt_w$. Then the nodes s and s' do the one-to-all personalized communication inside their own clusters. It takes $3t_s + (32+16+8)mt_w$. The nodes in the clusters of s and s' send a message of length 8 via cross-edge. It takes $t_s + 8mt_w$. Finally, the nodes do the one-to-all personalized communication inside their own clusters. It takes $3t_s + (4+2+1)mt_w$. The total time the one-to-all personalized communication takes is $8t_s + 127mt_w$.

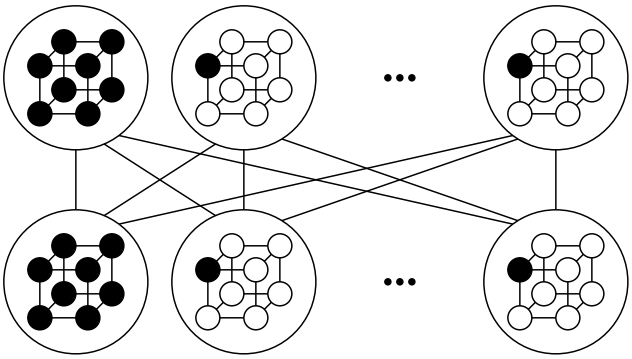
For the all-to-all broadcast, there are also 4 steps. First, each node broadcasts its message inside cluster. In this step, three data transmissions take place and the size of the message doubles after each transmission: (1) sending 1 message via dimension 2, (2) sending 2 messages via dimension 1, and (3) sending 4 messages via dimension 0. The sub-total



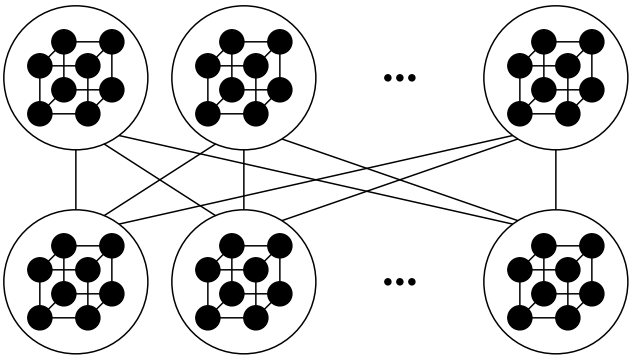
(a) Node 0 sends a message via cross edge



(b) Broadcast inside clusters



(c) Send the message via cross edge



(d) Broadcast inside clusters

Figure 2. One-to-all broadcast in $RDN^1(B)$

time it takes is $3t_s+(1+2+3)mt_w = 3t_s+7mt_w$. Then each node sends a message of length 8 via the cross-edge. This takes $t_s + 8mt_w$. Next, each node broadcasts its message received inside cluster. In this step, three data transmissions take place and the size of the message doubles after each transmission: (1) sending 8 messages via dimension 2, (2) sending 16 messages via dimension 1, and (3) sending 32 messages via dimension 0. The sub-total time it takes is $3t_s + (8+16+32)mt_w = 3t_s + 56mt_w$. Finally, each node sends a message of length 63 via the cross-edge. This takes $t_s + 63mt_w$. The total time the one-to-all broadcast takes is $(3t_s+7mt_w)+(t_s+8mt_w)+(3t_s+56mt_w)+(t_s+63mt_w) = 8t_s + 134mt_w$.

For the all-to-all personalized communication, there are also 4 steps. First, we do the personalized communication inside each cluster. A node contains 128 messages, each for node i, $0 \leq i \leq 127$. In this step, three data transmissions take place and the message size is 64. The sub-total time it takes is $3t_s+(64\times3)mt_w = 3t_s+192mt_w$. Then each node sends a message of length 64 via the cross-edge. This takes $t_s + 64mt_w$. Next, we do the personalized communication inside the clusters. In this step, three data transmissions take place. The sub-total time it takes is $3t_s + (64 \times 3)mt_w = 3t_s+192mt_w$. Finally, each node sends a message of length 64 via the cross-edge. This takes t_s+64mt_w. The total time the all-to-all personalized communication takes is $(3t_s + 192mt_w)+(t_s+64mt_w)+(3t_s+192mt_w)+(t_s+64mt_w) = 8t_s + 512mt_w$.

Now we consider an $RDN^2(B)$ that connects $2^{15} = 32,768$ nodes where B is a 3-cube.

For the one-to-all broadcast, source node sends the message via cross-edge of level 2. It takes $t_s + mt_w$. Then the nodes that have the message broadcast in $RDN^1(B)$. This takes $8t_s + 8mt_w$. Next, the nodes send the message via cross-edge of level 2. It takes $t_s + mt_w$. Finally, the nodes broadcast the message in $RDN^1(B)$. This takes $8t_s+8mt_w$. The total time the one-to-all broadcast takes is $18t_s+18mt_w$.

For the one-to-all personalized communication, the source node s send a packed message of length $(16384 - 128)m$ in words to the node s' via the cross-edge of level 2. Then nodes s and s' do the personalized communications in $RDN^1(B)$. Next, each node send a packed message of length $128m$ via cross-edge of level 2. Finally, the personalized communication is done in $RDN^1(B)$. The total time is $18t_s+(32767-128-8+1024+128-8+8)mt_w$.

For all-to-all broadcast in $RDN^2(B)$, we first do the all-to-all broadcast in $RDN^1(B)$. Then each node send a packed message of length $128m$ via cross-edge of level 2. Next, we do the all-to-all broadcast in $RDN^1(B)$. Finally, each node send a packed message of length $16383m$ via cross-edge of level 2. The total time is $18t_s + (32767 + 8 - 1 + 128 + 1024 - 1)mt_w$.

The all-to-all personalized communication is done similarly to the all-to-all broadcast. The total time is $18t_s +$

$16384 \times 18mt_w$.

Table II compares the performance of the algorithms in RDNs with those in the corresponding hypercubes. We can see that the communication times of the collective communications in RDNs are very close to those in hypercubes although there are more links in hypercubes than in RDNs.

Table II
COMMUNICATION TIMES

Pattern	7-cube	$RDN^1(B)$
One2All_B	$7t_s + 7mt_w$	$8t_s + 8mt_w$
One2All_P	$7t_s + 127mt_w$	$8t_s + 127mt_w$
All2All_B	$7t_s + 127mt_w$	$8t_s + 134mt_w$
All2All_P	$7t_s + 448mt_w$	$8t_s + 512mt_w$
Pattern	15-cube	$RDN^2(B)$
One2All_B	$15t_s + 15mt_w$	$18t_s + 18mt_w$
One2All_P	$15t_s + 32767mt_w$	$18t_s + 33783mt_w$
All2All_B	$15t_s + 32767mt_w$	$18t_s + 33925mt_w$
All2All_P	$15t_s + 245760mt_w$	$18t_s + 294912mt_w$

VII. CONCLUDING REMARKS

In this paper, we showed efficient routing algorithms for collective communications in a newly proposed network, RDN. The RDN is a very interesting and versatile interconnection network for MPPs. It is expected to be a potential candidate for high-performance interconnection of supercomputers of next generations. Collective communication lies at the center of parallel computing applications. Our results show that collective communication can be done efficiently in RDN. There are many other aspects of the RDN that can further widen the possible applications that might fit in the RDN and the parallel computer systems using RDN as its interconnection network. Some of the issues are listed below.

1) Develop efficient routing algorithms for other frequently used communication patterns such as multicast or fault-tolerant routing.
2) Evaluate the architecture complexity vs. performance of benchmarks.
3) Investigate the embedding of other frequently used topologies into a RDN.

REFERENCES

[1] S. G. Aki, *Parallel Computation: Models and Methods*. Prentice-Hall, 1997.

[2] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.

[3] A. Varma and C. S. Raghavendra, *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*. IEEE Computer Society Press, 1994.

[4] K. Ghose and K. R. Desai, "Hierarchical cubic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 4, pp. 427–435, April 1995.

[5] Y. Li and S. Peng, "Dual-cubes: a new interconnection network for high-performance computer clusters," in *Proceedings of the 2000 International Computer Symposium, Workshop on Computer Architecture*, ChiaYi, Taiwan, December 2000, pp. 51–57.

[6] Y. Li, S. Peng, and W. Chu, "Efficient collective communications in dual-cube," *The Journal of Supercomputing*, vol. 28, no. 1, pp. 71–90, April 2004.

[7] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Commun. ACM*, vol. 24, pp. 300–309, May 1981.

[8] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867–872, July 1988.

[9] G. H. Chen and D. R. Duh, "Topological properties, communication, and computation on wk-recursive networks," *Networks*, vol. 24, no. 6, pp. 303–317, 1994.

[10] G. Vicchia and C. Sanges, "A recursively scalable network vlsi implementation," *Future Generation Computer Systems*, vol. 4, no. 3, pp. 235–243, 1988.

[11] TOP500, *Supercomputer Sites*. http://top500.org/, Nov. 2009.

[12] P. Beckman, "Looking toward exascale computing, keynote speaker," in *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'08)*, University of Otago, Dunedin, New Zealand, December 2 2008.

[13] Redpaper, *Roadrunner: Hardware and Software Overview*. http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf: IBM Corporation, 2009.

[14] Y. Li, S. Peng, and W. Chu, "Recursive dual-net: A new universal network for supercomputers of the next generation," in *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'09)*. Taipei, Taiwan: Springer, Lecture Notes in Computer Science (LNCS), June 2009, pp. 809–820.

[15] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks: an engineering approach*. IEEE Computer Society Press, 1997.

[16] S. L. Johnson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Transactions on Computers*, vol. 38, no. 9, pp. 1249–1268, 1989.

[17] Y. Lan, A. H. Esfahanian, and L. M. Ni, "Multicast in hypercube multiprocessors," *Journal of Parallel and Distributed Computing*, vol. 16, no. 1, pp. 30–41, 1990.

[18] P. K. McKinley, Y. J. Tsai, and D. Robinson, "Collective communication in wormhole-routed massively parallel computers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 2, pp. 184–190, 1996.

[19] J. G. Peters and M. Syska, "Circuit-switched broadcasting in torus networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 3, pp. 246–255, 1996.

[20] A. M. Farley, "Minimum-time broadcast networks," *Networks*, vol. 10, pp. 59–70, 1980.

[21] P. Kermani and L. Kleinrock, "Virtual cut-through: a new communication switching technique," *Computer Networks*, vol. 13, pp. 267–286, 1979.

[22] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Press, 1994.